An introduction to statistical parametric speech synthesis

Sādhanā Vol. 36, Part 5, October 2011, pp. 837–852. © Indian Academy of Sciences

An introduction to statistical parametric speech synthesis

SIMON KING

The Centre for Speech Technology Research, University of Edinburgh, Edinburgh, UK
e-mail: simon.king@ed.ac.uk

Abstract. Statistical parametric speech synthesis, based on hidden Markov model-like models, has become competitive with established concatenative techniques over the last few years. This paper offers a non-mathematical introduction to this method of speech synthesis. It is intended to be complementary to the wide range of excellent technical publications already available. Rather than offer a comprehensive literature review, this paper instead gives a small number of carefully chosen references which are good starting points for further reading.

Keywords. Speech synthesis; hidden Markov model-based speech synthesis; statistical parametric speech synthesis; vocoding; text-to-speech.

1. Introduction

This tutorial paper provides an overview of statistical parametric approaches to text-to-speech synthesis. These approaches are often called simply 'HMM synthesis' because they generally use Hidden Markov Models, or closely related models. This tutorial attempts to concisely explain the main concepts of this approach to speech synthesis without becoming lost in technical detail. Some basic familiarity with HMMs is assumed, to the level covered in Chapter 9 of Jurafsky and Martin (2009).

2. Text-to-speech synthesis

The automatic conversion of written to spoken language is commonly called 'text-to-speech' or simply 'TTS'. The input is text and the output is a speech waveform. A TTS system is almost always divided into two main parts. The first of these converts text into what we will call a 'linguistic specification' and the second part uses that specification to generate a waveform. This division of the TTS system into these two parts makes a lot of sense both theoretically and for practical implementation: the front end is typically language-specific, whilst the waveform generation component can be largely independent of the language (apart from the data it contains, or is trained on).

The conversion of text into a linguistic specification is generally achieved using a sequence of separate processes and a variety of internal intermediate representations. Together, these are known as the 'front end'. The front end is distinct from the 'waveform generation' component which produces speech, given that linguistic specification as input.

The focus in this tutorial is on speech synthesis using statistical parametric methods. For a more general discussion of speech synthesis, Taylor (2009) is recommended as follow-up reading. The best starting point for further reading on statistical parametric synthesis is Zen *et al* (2009), followed by Zen *et al* (2007). A comprehensive bibliography can be found online at http://hts.sp.nitech.ac.jp/?Publications and a list of the available resources for experimenting with statistical parametric speech synthesis can be found in Zen and Tokuda (2009); many of them are free.

In this tutorial, I have taken the decision to provide only a very short bibliography, in order to make it easier for the reader to select suitable follow-up reading.

3. From vocoding to synthesis

Descriptions of speech synthesisers often take a procedural view: they describe the sequence of processes required to convert text into speech, often arranged in a simple 'pipeline' architecture. But another way to think about speech synthesis is to start from the idea of vocoding, in which a speech signal is converted into some (usually more compact) representation so that it can be transmitted. A vocoder looks like figure 1. We can think about speech synthesis in a similar framework but, instead of transmitting the parameterized speech, it is stored (figure 2). We can later retrieve the parameterization and proceed to generate the corresponding speech waveform.

Such a system has two distinct phases, which we can call 'training' and 'synthesis'. In the training phase, the stored form is acquired from a speech corpus (the 'training data'). By indexing this stored form with a linguistic specification, it will be possible to perform synthesis with only this linguistic specification as input, obtaining a speech waveform as output.

The stored form can be either the speech data itself, or a statistical model derived from the data. Whilst these appear to be quite different approaches to speech synthesis, thinking about them

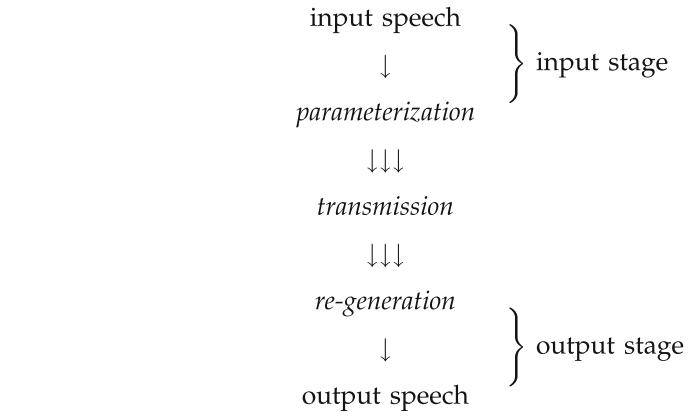

Figure 1. A vocoder. The multiple arrows indicate that the parameterized representation of speech typically has several distinct groups of parameters – e.g., filter coefficients capturing the spectral envelope and source parameters such as the fundamental frequency F0.

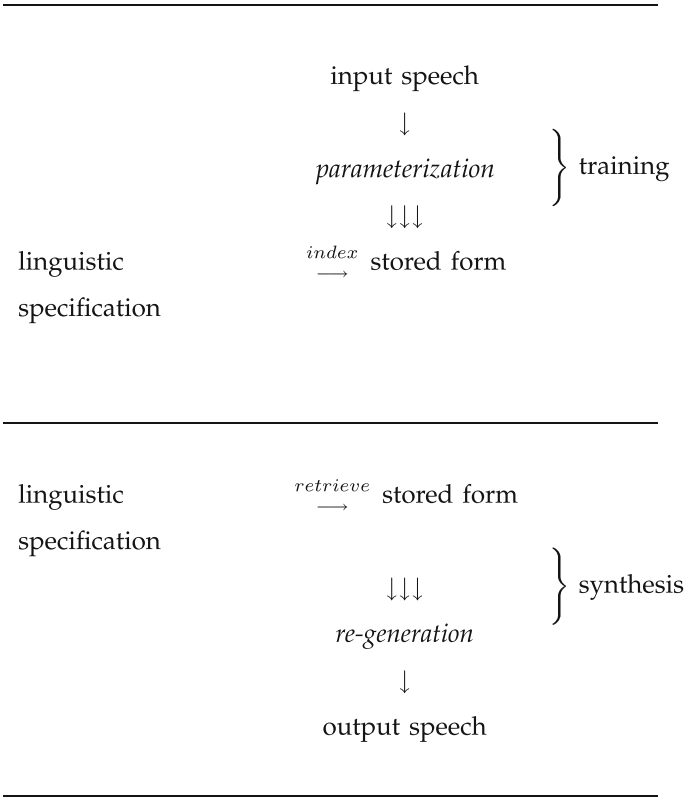

Figure 2. Speech synthesis, viewed as a vocoder. The input stage of the vocoder has become 'training' and is performed just once for the entire speech corpus (training data). The output stage of the vocoder has become 'synthesis', which is performed once for each novel sentence to be synthesized.

together, in a common vocoding-like framework, will give us some insight into the relationship between them.

3.1 *The linguistic specification*

In the synthesis phase described above, the input is a linguistic specification. This could be as simple as a phoneme sequence, but for better results it will need to include supra-segmental information such as the prosody pattern of the speech to be produced. In other words, the linguistic specification comprises whatever factors might affect the acoustic realisation of the speech sounds making up the utterance.

One way to think about the linguistic specification is to focus on a particular speech sound: consider the vowel in the word 'speech' as an example. The linguistic specification must capture all of the information that could affect how this vowel sounds. In other words, it is a summary of all of the information in the *context* in which this vowel appears. In this example, important contextual factors will include the preceding bilabial unvoiced plosive (because that will influence the formant trajectories in the vowel) and the fact that the vowel is in a mono-syllabic word (because that will influence the duration of the vowel, amongst other things); many other factors will also have some effect, to varying degrees, on this vowel.

Table 1. An example list of context factors which could comprise the linguistic specification.

Preceding and following phonemes
Position of segment in syllable
Position of syllable in word & phrase
Position of word in phrase
Stress/accent/length features of current/preceding/following syllables
Distance from stressed/accented syllable
POS of current/preceding/following word
Length of current/preceding/following phrase
End tone of phrase
Length of utterance measured in syllables/words/phrases

The context will of course include factors within the same word and the same utterance, such as the surrounding phonemes, words and the prosodic pattern, but could also extend to the surrounding utterances and then on to paralinguistic factors such as the speaker's mood or the identity of the listener. In a dialogue, the context may need to include factors relating to the other speaker. However, for practical reasons, only factors within an utterance are considered by most current systems. Table 1 lists the context factors that may be considered in a typical system.

This list of factors that have the potential to influence each speech sound is rather long. When we consider the number of different values that each one may take (e.g., preceding phoneme could take up to 50 different values), and find the number of permutations, it quickly becomes clear that the number of different contexts is vast, even if we only consider linguistically-possible combinations. But not all factors will have an effect all of the time. In fact, we hope that only a few factors will have any significant effect at any given moment. This reduces the number of effectively different contexts down to a more manageable number. The key problem, which we will re-visit in section 5, is to determine which factors matter and when.

For each novel sentence to be synthesized, it is the task of the front end to predict the linguistic specification from text. Inevitably, many tasks performed by the front end (e.g., predicting pronunciation from spelling) are quite specific to one language or one family of languages (e.g., those with alphabetic writing systems). A discussion of the front end is out of scope of the current paper, but coverage of this topic can be found in Taylor (2009).

3.2 *Exemplar-based systems*

An exemplar-based speech synthesis system simply stores the speech corpus itself; that might mean the entire corpus or just selected parts of it (e.g., one instance of each type of speech sound from a set of limited size). Indexing this stored form using a linguistic specification means labelling the stored speech data such that appropriate parts of it can be found, extracted then concatenated during the synthesis phase. The index is used just like the index of a book – to 'look up' all the occurrences of a particular linguistic specification. In a typical unit selection system, the labelling will comprise both aligned phonetic and prosodic information. The process of retrieval is not entirely trivial, since the exact specification required at synthesis time may not be available in the corpus, so a *selection* must be performed to choose, from amongst the many slightly mismatched units, the best available sequence of units to concatenate. The speech may be

stored as waveforms or in some other representation more suitable for concatenation (and small amounts of signal modification) such as residual-excited linear production coefficients (LPC).

3.3 *Model-based systems*

A model-based system does not store any speech. Instead, it fits a model to the speech corpus during the training phase and stores this model. The model will typically be constructed in terms of individual speech units, such as context-dependent phonemes: the model is thus indexed by a linguistic specification. At synthesis time, an appropriate sequence of context-dependent models is retrieved and used to generate speech. Again, this may not be trivial because some models will be missing, due to the finite amount of training data available. It is therefore necessary to be able to create – 'on the fly' – a model for any required linguistic specification. This is achieved by sharing parameters with sufficiently similar models – a process analogous to the selection of slightly mis-matched units in an exemplar-based system.

3.4 *Indexing the stored form*

In order for the stored form – whether it is speech or a model – to be indexed by the linguistic specification, it is necessary to produce the linguistic specification for every utterance in the speech corpus (training data). Manual labelling is one way to achieve this, but that is often impractical or too expensive. The most common approach is to use the same front end that will be used when synthesizing novel sentences, to predict the linguistic specification based on the text corresponding to the speech corpus. This is unlikely to be a perfect match to what the speaker actually said. However, a few simple techniques, based on forced alignment methods borrowed from automatic speech recognition, can be applied to improve the accuracy of this labelling, including automatically identifying the true pause locations and some of the pronunciation variation.

4. Statistical parametric models for speech synthesis

When we talk about a model-based approach to speech synthesis, particularly when we wish to learn this model from data, we generally mean a statistical parametric model. The model is *parametric* because it describes the speech using parameters, rather than stored exemplars. It is *statistical* because it describes those parameters using statistics (e.g., means and variances of probability density functions) which capture the distribution of parameter values found in the training data. The remainder of this article will focus on this method for speech synthesis.

Historically, the starting point for statistical parametric speech synthesis was the success of the HMM for automatic speech recognition. No-one would claim that the HMM is a true model of speech. But the availability of effective and efficient learning algorithms (Expectation–Maximization), automatic methods for model complexity control (parameter tying) and computationally efficient search algorithms (Viterbi search) make the HMM a powerful model. The performance of the model, which in speech recognition is measured using word error rates and in speech synthesis by listening tests, depends critically on choosing an appropriate configuration. The two most important aspects of this configuration are the parameterization of the speech signal (the 'observations' of the model, in HMM terminology) and the choice of modelling unit. Since the modelling unit is typically a context-dependent phoneme, this choice means

Table 2. Comparison of Hidden (Semi) Markov Model configurations for recognition vs. synthesis.

	Recognition	Synthesis
observations	spectral envelope represented using around 12 parameters	spectral envelope represented using 40–60 parameters, plus source features
modelling unit	triphone, considering preceding and following phoneme	full context, considering preceding two and succeeding two phonemes plus all other context features listed in table 1
duration model	state self-transitions	explicit parametric model of state duration
parameter estimation	Baum–Welch	Baum–Welch, or trajectory training
decoding	Viterbi search	not usually required
generation	not required	Maximum-likelihood parameter generation

selecting which contextual factors need to be taken into account. Table 2 summarizes some differences in the configuration of models for automatic speech recognition and speech synthesis.

4.1 *Signal representation*

The speech signal is represented as a set of vocoder parameters at some fixed frame rate. A typical representation might use between 40 and 60 parameters per frame to represent the spectral envelope, the value for F0 (the fundamental frequency), and 5 parameters to describe the spectral envelope of the aperiodic excitation. Before training the models, the encoding stage of the vocoder is used to extract a vector comprising these vocoder parameters from the speech signal, at a frame rate of typically 5 ms. In the synthesis phase, the entire vector is generated by the models, then used to drive the output stage of the vocoder.

In principle, any vocoder could be used for HMM-based speech synthesis, provided that the parameters it uses are sufficient to reconstruct the speech signal with high quality and that these parameters can be automatically extracted from speech in the training phase. It could even be something like a formant synthesizer. However, since the parameters will be statistically modelled, some vocoders will offer better performance than others. The fundamental operations that occur in the statistical modelling are the averaging of vocoder parameters during the training phase, and the generation of novel values (we can liken this to interpolation and extrapolation of the values found in the training data) during the synthesis phase. So, the vocoder parameters must be well-behaved under such operations and not lead to unstable values. For example, line spectral pairs would probably be a better representation than linear prediction co-efficients, because the former are well-behaved under interpolation whereas the latter can result in a unstable filter.

A popular vocoder used widely in HMM synthesis is called STRAIGHT (Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum). A full description of this channel vocoder is beyond the scope of this paper and can instead be found in Kawahara *et al* (1999). For our purposes, it is sufficient to state that STRAIGHT possesses the desired properties described above and performs very well in practice.

4.2 *Terminology*

4.2a *Not HMMs but HSMMs*: The model most commonly used in statistical parametric speech synthesis is not in fact the HMM at all. The duration model (i.e., the state self-transitions) in the HMM is rather simplistic and a better model of duration is required for high-quality speech

synthesis. Once an explicit duration model is added to the HMM, it is no longer a Markov model. The model is now only 'semi-Markov' – transitions between states still exist, and the model is Markov at that level, but the explicit model of duration within each state is not Markov. The model is now a Hidden Semi-Markov Model, or HSMM. Wherever we talk about HMM speech synthesis, we most often really mean HSMM speech synthesis.

4.2b *Labels and context*: The linguistic specification described earlier is a complex, structured representation; it may comprise lists, trees and other linguistically-useful structures. HMM-based speech synthesis involves generating speech from a linear sequence of models, in which each model corresponds to a particular linguistic unit type. Therefore, it is necessary to flatten the structured linguistic specification into a linear sequence of *labels*. This is achieved by attaching all the other linguistic information (about syllable structure, prosody, etc.) to the phoneme tier in the linguistic specification – the result is a linear sequence of context-dependent phonemes. Given these *full context labels*, the corresponding sequence of HMMs can be found, from which speech can be generated.

4.2c *Statics, deltas and delta–deltas*: The vocoder parameters themselves are the only thing required to drive the output stage of the vocoder and produce speech. However, the key to generating natural-sounding speech using HMM (or HSMM) synthesis lies in the modelling not only of the statistical distribution of these parameters, but in also modelling their rate of change – i.e., their velocity – as described in section 4.4. Following terminology borrowed from automatic speech recognition, the vocoder parameters are known as the *static coefficients*, and their first-order derivatives are known as the *delta coefficients*. In fact, further benefit can be gained from modelling acceleration as well, thus we have *delta–delta coefficients*.

These three types of parameters are stacked together into a single observation vector for the model. During training, the model learns the distributions of these parameters. During synthesis, the model generates parameter trajectories which have appropriate statistical properties.

4.3 *Training*

Just as in automatic speech recognition, the models for HMM synthesis must be trained on labelled data. The labels must be the full context labels described above and they are produced for the training data in the way described in section 3.4.

4.4 *Synthesis*

The process of synthesis – actually generating speech, given only text as input – proceeds as follows. First, the input text is analysed and a sequence of full context labels is produced. The sequence of models corresponding to this sequence of labels is then joined together into a single long chain of states. From this model, the vocoder parameters are generated using the MLPG algorithm outlined below. Finally, the generated vocoder parameters are used to drive the output stage of the vocoder to produce a speech waveform.

Generating the parameters from the model: The principle of maximum likelihood (i.e., the same criterion with which the model is usually trained) is used to generate a sequence of observations (vocoder parameters) from the model. First, we will consider doing this in a naive way and see that this produces unnatural parameter trajectories. Then we will introduce the method actually

used. Note that the term 'parameter' is being used to refer to the output of the model, and not the *model* parameters (the means and variances of the Gaussian distributions, etc.).

Duration: In both the naive method and the maximum likelihood parameter generation (MLPG) algorithm described next, the durations (i.e., the number of frames of parameters to be generated by each state of the model) are determined in advance – they are simply the means of the explicit state duration distributions.

Naive method for parameter generation: This method generates the most likely observation from each state, considering only the static parameters. The most likely observation is, of course, the mean of the Gaussian in that state. So this method generates piecewise constant parameter trajectories, which change value abruptly at each state transition. Clearly, this will not sound natural when used to drive the vocoder: natural speech does not have parameter trajectories that look like this. This problem is solved by the MLPG algorithm.

The maximum likelihood parameter generation algorithm: The naive method above has missed once crucial aspect of the parameter trajectories we find in natural speech. It only considered the statistical properties of the *static* parameters. But in natural speech it is not only the absolute value of the vocoder parameters that behave in a certain way, it is also the speed with which they change value. We must therefore also take the statistical properties of the *delta* coefficients into account. In fact, we can also consider the statistical properties of the *delta–delta* coefficients.

Figure 3 illustrates the MLPG algorithm. The HMM has already been constructed: it is the concatenation of the models corresponding to the full context label sequence, which itself has been predicted from text by the front end. Before generating parameters, a state sequence is

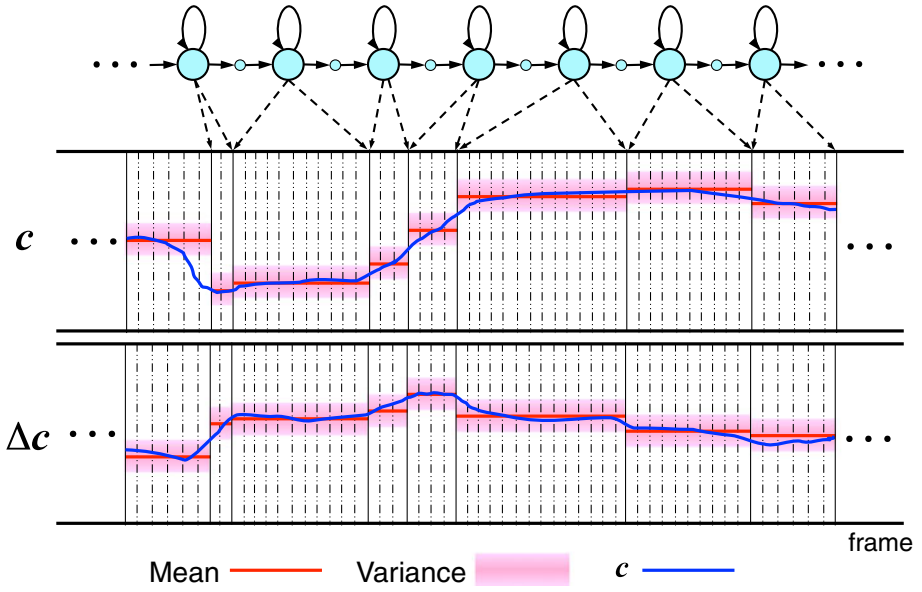

Figure 3. Maximum likelihood parameter generation: a smooth trajectory for parameter c is generated from a discrete sequence of distributions over c, by taking the distribution over the delta (and delta–delta) coefficients of c into account. From http://hts.sp.nitech.ac.jp/archives/2.2beta/HTS_Slides.zip.

chosen using the duration model. This determines how many frames will be generated from each state in the model. The figure shows the sequence of output distributions for each state, frame by frame. MLPG finds the most likely sequence of generated parameters, given the distributions for the static, delta and delta–delta distributions. The figure shows these for only the 0^{th} cepstral coefficient $c(0)$, but the principle is the same for all the parameters generated by the model, such as F0.

The easiest way to understand what MLPG achieves is to consider an example: in the figure, locate a region where $\Delta c(0)$ is positive: the static parameter $c(0)$ is rising at that point – it has a positive slope. So, even though the statistical properties of the static coefficients are piecewise constant, the most likely parameter trajectory is constantly and smoothly changing in a statistically appropriate way. For a more complete explanation of MLPG, including a mathematical treatment, refer to Zen *et al* (2009).

5. Generating novel utterances: the challenge of unseen contexts

To state the obvious: a key problem in speech synthesis is generating utterances that we do not have natural recordings of. This necessarily involves building the utterance from smaller units (whether by concatenation or generation from a model). Since we will generally not have seen instances of those unit types in exactly the same context before, the problem can be stated in terms of generalizing from the limited set of contexts that we observe in the training data, to the almost unlimited number of unseen contexts which we will encounter when synthesizing novel utterances.

But just how often will this happen? If our speech corpus is large, surely it will cover all of the commonly-occurring contexts? Unfortunately, this is not the case.

One obvious cause is the very rich context suggested in table 1, which has two consequences. First, since the context spans the entire utterance, each occurrence of a context-dependent unit in the speech corpus will almost certainly be unique: it will occur only once (assuming we do not have any repeated sentences in our corpus). Second, the vast majority of possible context-dependent units will never occur in the corpus: the corpus has a very *sparse* coverage of the language.

Even forgetting for a moment this rich context-dependency, the distribution of *any* linguistic unit (e.g., phonemes, syllables, words) in a corpus will be far from uniform. It will have a long tail of low- or zero-frequency types. In other words, there will be very many types of unit which occur just once or not at all in our corpus. This phenomenon is known as the 'large number of rare events'. Each type is itself very rare, but since there are so many different types, we are still very likely to encounter one of them. What this means for speech synthesis is that, for any sentence to be synthesized, there is a high probability that some very rare unit type (e.g., context-dependent phoneme) will be needed. No finite speech corpus can contain all of the rare types we might need, so simply increasing the corpus size could never entirely solve this problem.

That being the case, some other solution is required. We can now clearly see the problem as one of generalizing from limited training data, which leads to a solution in the form of *model complexity control*.

5.1 *Generalization*

Generalization, from a limited number of observed examples to a potentially unlimited number of unseen contexts is a widely-encountered problem in most applications of machine learning.

A typical situation, especially in natural language applications, is the long-tailed distribution mentioned above (often likened to a Zipf distribution). In other words, a few types will have many examples in the data, but most will have few or no examples. This makes it impossible to directly model rare or unobserved types – there are simply not enough examples of them to learn anything. This is certainly the situation in speech synthesis, where the types are phonemes-in-context.

Reducing the number of types with which we label the data will mitigate this problem. In speech synthesis, that would mean reducing the number of context factors considered. But, we do not know *a priori* which context factors could be removed and which ones must be kept because they have a strong effect on the realization of the phoneme in question. Moreover, precisely which context factors matter will vary they also interact in complex combinations.

An elegant solution is to continue to use a large number of *types* to label the data and to control the complexity of the model rather than the complexity of the labelling. The method for controlling model complexity most commonly used in HMM-based synthesis is borrowed from automatic speech recognition and involves sharing (or 'tying') parameters amongst *similar* models in order to achieve:

1) appropriate model complexity (i.e., the right number of free parameters for the data available);
2) better estimates of parameters for which we have seen only a few examples;
3) estimates of parameters for which we have seen no examples at all.

In order to decide which models are sufficiently similar (so can share parameters), consider again those contextual factors. Since (we hope that) only a few factors need to be taken into consideration at any given moment, we can aim for a set of context-dependent models in which, for each model, only the relevant context is taken into account. The amount of context-dependency may therefore differ from model to model. The result is that only those context distinctions supported by evidence in the data are modelled. Context factors which have no effect (at least, none that we can discern in the data) are dropped, on a model-by-model basis. As a simple example, imagine that the identity of the preceding phoneme has no discernible effect on the realization of [ʃ] but that the identity of the following phoneme does have some effect. If that were the case, groups of models could share the same parameters as follows: there would be a single model of [ʃ] for all of the contexts [...aʃt...], [...ɪʃt...], [...ɛʃt...], etc. and another single model for all of [...aʃə...], [...ɪʃə...], [...ɛʃə...], etc., and so on.

The mechanism for deciding which models can be shared across contexts is driven by the data. The complexity of the model (i.e., how much or how little parameter tying there is) is automatically chosen to suit the amount of training data available: more data will lead to a more complex model.

5.2 *Model complexity control using parameter tying*

Complexity control means choosing the right number of model-free parameters to suit the amount of training data available. In HMM-based speech synthesis, this means in effect choosing which contextual distinctions are worth making and which do not matter. In other words, when should we use separate models for two different contexts and when should we use the same model?

A widely used technique for model complexity control in automatic speech recognition involves clustering together similar models. The possible ways in which the models could be

grouped into clusters are specified in terms of context factors, and the actual clustering that is chosen is the one that gives the best fit of the model to the training data. This method has been adopted by HMM-based speech synthesis, where it is even more important than in speech recognition simply because there is a much larger number of different contexts to deal with. A description of the decision tree method for clustering can be found in section 10.3 of Jurafsky and Martin (2009).

After the models have been clustered, the number of distinct models is much lower than the number of distinct contexts. The clustering has automatically discovered the optimal number of context distinctions that can be made, given the available data. With a larger training data set, we would be able to use a larger number of models and make more fine-grained distinctions.

Note that, in practice, it is individual states and parameters that are tied rather than whole models. However, the principle is the same.

5.3 *Relationship to unit selection*

In unit selection synthesis, the effect of the context factors on the choice of unit is measured by the target cost. The most common form of function for the target cost is a simple weighted sum of penalties, one for each mismatching context factor. The target cost aims to identify the 'least bad' unit candidates from the database. An alternative form of target cost called 'clunits' uses a context clustering tree in a similar fashion to the model clustering method described above, but where the leaves of the tree contain not model parameters but clusters of speech units from the database.

The goal is the same: to generalize from the data we have seen, to unseen contexts. This is achieved by automatically finding which contexts are in effect interchangeable. In unit selection that means finding a group of sufficiently similar candidate units to use in a target context that is missing from the speech corpus; in HMM synthesis it means averaging together speech from groups of contexts to train a single model.

5.4 *Where next?*

Speech synthesis has progressed from the use of simple models of the vocal tract in terms of formant frequencies and bandwidths etc., driven by rules – as in the well-known Klatt vocal tract model and the various synthesisers it has been used it such as MITalk and DECTalk – through the use of diphone concatenation and unit selection, and most recently to the use of statistical parametric models. This latest use of models differs from the earlier one in that the 'model' of the speech signal is that of a vocoder and the rules have been replaced with probability distributions learned from data. But statistical parametric speech synthesis could benefit from a stronger model of the speech signal, with perhaps a more explicit representation of the physical and linguistic constraints of speech. This may be an interesting future direction.

6. Some frequently-asked questions about statistical parametric speech synthesis

6.1 *How is prosody predicted?*

There are two parts to the answer here. First, a symbolic representation of prosody is predicted by the front-end, in just the same way as for concatenative synthesis. Second, this symbolic representation is used as part of the context factors in the *full context models* used to generate speech.

Provided that (i) there are sufficient training examples of each prosodic context and (ii) there is some consistent relationship between the prosodic labelling and the true prosody of the training data, then there will be different models for each distinct prosodic context and the models will generate appropriate prosody when used for speech synthesis. If either of (i) or (ii) are not true, then the parameter clustering will not be able to form models which are specific to particular prosodic contexts.

6.2 *What causes the 'buzzy' quality of the synthetic speech?*

This is because the speech is vocoded. The buzziness is mainly due to an over-simplistic model of the voice source. Vocoders which use *mixed excitation* – that is, they mix both periodic and aperiodic sources rather than switching between them – sound considerably less buzzy.

6.3 *What causes the 'muffled' quality of the synthetic speech?*

Averaging, which is an inevitable process in the training of the statistical model, can cause the speech to sound muffled. Averaging together many frames of speech, each with slightly differing spectral properties, will have the effect of widening the formant bandwidths and reducing the dynamic range of the spectral envelope. Likewise, averaging tends to produce over-smooth spectral envelopes and over-smooth trajectories when synthesising. One popular way to counter-act these effects is to, in effect, adjust the generated parameters so that they have the same variance found in natural speech. This method is called global variance (GV).

6.4 *Why is duration modelled separately?*

The model of duration in a standard HMM arises from the self transitions on each state. Under such a model, the most likely duration is always just one frame per state, which is obviously not correct for natural speech. Therefore, an explicit duration model is necessary. The model of duration is not really separate from the model of the spectral envelope and source. They interact through the model structure. However, the context factors which affect duration will differ from those which affect the spectrum and source features, so these various groups of model parameters are clustered separately.

6.5 *Do you really need so much context information in the labels, given how much clustering takes place?*

It's certainly true that many context factors will not be relevant most of the time. This is a good thing, because it means the number of effectively distinct contexts is far smaller than the number theoretically possible from the combinations of context factor values. One great advantage of the decision tree method for clustering is that it will only use those context factors which correspond to model-able acoustic distinctions in the data. It therefore does no harm to include factors which might not actually matter: decision tree clustering will only use them if they do correspond to an acoustic distinction. If in doubt, we tend to include all possible context factors, then let the decision tree clustering identify the useful ones.

6.6 *How much training data do you need before it starts being competitive with state-of-the-art concatenative methods?*

In fact, statistical parametric synthesis probably has more advantage over concatenative methods on small datasets than large ones. With very large and highly consistent datasets, well-engineering concatenative systems are capable of producing excellent quality synthetic speech. However, on smaller or less ideal data, statistical parametric synthesis tends to have an advantage because the statistical approach (in particular, the process of averaging inherent in fitting the model to the data) can 'iron out' inconsistencies in the data which concatenative methods are overly sensitive to.

6.7 *What is the effect of the amount of training data? Why does more data usually lead to a better-sounding synthesizer?*

A nice property of the statistical approach is that the quality of the resulting voice scales up as the amount of data is increased. This is because a larger database will not only contain more speech, but perhaps even more importantly it will contain a wider variety of contexts. This will lead to larger parameter tying trees and thus to more fine-grained – that is, more context-sensitive – models, which will in turn produce better speech. This process of scaling up the complexity of the model (section 2) is entirely automatic. As we add more data, we get more complex models.

6.8 *How important is the quality of the recordings used for training the system?*

'Quality' can mean various things, but in general statistical parametric systems can produce better results than concatenative systems can for low quality data. Great care must be taken in recording the data for a concatenative system, in terms of recording quality and speaker consistency. Whilst such data is the ideal starting point for statistical parametric synthesis too, this method can – if necessary – use less ideal data (e.g., 'found' speech such as podcasts or radio news broadcasts).

6.9 *How small can the system be?*

The training time of the system will of course increase with the amount of data used, but training time is not usually a critical factor since it happens only once.

The size of the system – that is, how much memory or disk space it needs – can actually be scaled up or down quite simply by varying the size of the parameter clustering decision trees. A smaller tree has fewer leaves and therefore corresponds to a model with fewer parameters. The quality of the resulting synthetic speech will of course vary, and the tradeoff between memory and quality can be chosen according to the application.

6.10 *How are the dependencies between duration, excitation and spectral parts modeled?*

There are no explicit model parameters concerned with modelling the dependencies between these various aspects of the model. They are modelled separately. However, all model parameters are context-dependent, so if there are systematic patterns of covariance between F0 and duration (for example), these can be captured by the context-dependent modelling, provided that the covariations are predictable from the context.

6.11 *Why does the Global Variance technique make the speech sound so much better, and are there any drawbacks?*

Global Variance is a popular way to address the reduced dynamic range and overly smooth characteristics of the generated vocoder parameters. It gives the output speech more natural characteristics; for example, speech generated using GV will typically have a spectral envelope with sharper formant peaks than speech generated without GV – this is why GV can dramatically reduce the 'muffled' effect.

One drawback of GV is that it can cause artefacts, particularly for short utterances, because it essentially adjusts the variance of the generated speech parameters to always have a particular fixed variance. However, short utterances will naturally have less variance in their vocoder parameters than longer ones, simply because they contain less variety of phonetic segments; by forcing their variance to match the global variance, extreme values for the vocoder parameters can be produced, which result in artefacts in the output speech.

GV is seen by some as a post-hoc fix for a problem caused by the statistical modelling. However, most researchers agree that it is highly effective.

6.12 *How should one evaluate the quality of HMM-based speech synthesis? How does HMM based synthesis perform in comparison with unit selection or hybrid based synthesis?*

There is no short answer to the question of how to evaluate speech synthesis, but a good starting point would be to read the evaluation reports from the Blizzard Challenge, in which a conventional approach is taken to evaluating naturalness and intelligibility using large-scale listening tests. The Blizzard Challenge is also a good place to find comparisons of both synthesis techniques. In simple terms, it is generally found that HMM-based speech synthesis is more intelligible but less natural-sounding than unit selection. Hybrid systems are amongst the most natural-sounding available, but are still not as intelligible as HMM-based synthesis. However, this is a rapidly moving field and this answer may be out-of-date by the time you read this!

6.13 *How language-dependent is the decision tree clustering for context-dependent models?*

The method of clustering is entirely independent of the language being used. The range of possible questions which could appear in the tree is language-dependent to some degree, since not all languages share the same set of linguistic features. However, there is a large overlap between languages: for example, context-dependent phonemes work well for many languages, so questions about the left and right phonetic context are often a good choice. The tree itself is learned from the training data and will vary not only from one language to another, but also from one dataset to another. But again, there are many common features: for example, in the tree for clustering spectral parameters, questions about the left and right phonetic context will typically appear near the root of the tree, whereas in the tree for clustering the F0 parameters, the questions near the tree root are more likely to be about prosodic features such as phrase boundaries and pitch accents.

6.14 *Is it possible to do synthesis with context-independent models?*

Yes, it is possible to generate speech from such models, but it sounds substantially worse than when using context-dependent models.

6.15 *Apart from STRAIGHT, what other vocoding techniques are used in HMM-based synthesis?*

Almost any vocoder can be employed in HMM-based speech synthesis, provided that it separates source and filter (so they can be modelled using separate parameters). However, not all parameterizations of speech are suitable for statistical modelling. For example, linear prediction coefficients (LPC) would be a poor choice because filter stability cannot be guaranteed and the models could in theory generate LPCs leading to unstable filters. Line spectral frequencies are a better choice, for this reason. Vocoding is not a solved problem: the development of vocoders specifically matched to the behaviour of the statistical model could lead to improved quality beyond that possible using current vocoders like STRAIGHT.

6.16 *In unit selection, the units can be of varying size. Could varying-size units be used in HMM-based synthesis?*

This is theoretically possible but the motivations for doing so are not as obvious as for unit selection. The goal of varying-size units in concatenative synthesis is to capture important context effects and to avoid making joins in difficult places. HMM-based synthesis captures context effects in a different way, and does not make joins. In specific situations, the use of units other than context-dependent phonemes may have advantages – see the question about tone languages.

6.17 *What are the possible other applications of HMM-based synthesis?*

HMM-based synthesis has been used to drive animation including talking heads and body motion, amongst other applications.

6.18 *Do any modifications need to be made for tone languages such as Chinese?*

Since tones extend over more than one phoneme, it may be appropriate to use larger units for such languages. In Chinese, so-called 'initial-final' units are often used. However, it is also possible to use phoneme units because context-dependent modelling (in which one of the context factors is the tone) can account for this phenomenon.

My understanding of statistical parametric speech synthesis was largely gained through interactions with Junichi Yamagishi in Edinburgh and Keiichi Tokuda and his group in Nagoya. Credit them with everything that is correct; blame me for any errors. I am also grateful to students in Edinburgh for suggesting some of the frequently asked questions.

References

Jurafsky D, Martin J H 2009 *Speech and language processing*, 2nd edition (Upper Saddle River, New Jersey, USA: Prentice Hall)

Kawahara H, Masuda-Katsuse I, de Cheveigné A 1999 Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds, *Speech Commun.* 27(3–4): 187–207

Taylor P 2009 *Text-to-speech synthesis* (Cambridge: Cambridge University Press)

Zen H, Tokuda K 2009 TechWare: HMM-based speech synthesis resources, IEEE Signal Processing Magazine

Zen H, Tokuda K, Black A W 2009 Statistical parametric speech synthesis, *Speech Commun.* 51(11): 1039–1064

Zen H, Tokuda K, Kitamura T 2007 Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences, *Comput. Speech Lang.* 21(1): 153–173